**Engenharia de Software** 2023-01

| **Professor(a)(es)** | RADAMÉS PEREIRA |
| --- | --- |

Atributos de uma boa Especificação de Requisitos de Projeto de Software:

● Clareza

● Não Ambígua

● Completa

● Simples

● Bem escrita

**UNIVERSIDADE COMUNITÁRIA REGIONAL DE CHAPECÓ - UNOCHAPECÓ**

**ÁREA DE CIÊNCIAS EXATAS E AMBIENTAIS**

**CIÊNCIA DA COMPUTAÇÃO**

Título

Sistema para controlar a produção da indústria de Concreto e suas vendas

Equipe (engenheiro e desenvolvedores): **Caleb Foreste**

**…**

**Introdução**

***1.1 Resumo do Projeto (Descrição textual)***

O projeto é o desenvolvimento de um sistema que controla uma fábrica de concreto que inclui vendas, escritório, e uma planta industrial.

***1.2 Plataforma de desenvolvimento (O equipamento dos desenvolvedores e ferramentas de software)***

Ferramenta utilizada até agora para o desenvolvimento é o visual studio code com plantuml.

***1.3 Plataforma de operação (O equipamento do cliente/usuário do sistema)***

Descreve-se aqui uma primeira visão das tecnologias para operacionalização.

***1.4 Definições e siglas (quaisquer siglas utilizadas no domínio, do vocabulário do usuário)***

Descreve-se aqui a definição de todas as siglas, abreviações e termos usados.

***1.5 Perspectiva do produto***

**1.5.1 Modos de operação (Meios de acesso ao**

**sistema/Arquitetura do sistema)**

Identificam-se aqui os modos requeridos de operação, tais como: Back

End-Front-End, Móvel, Stand-Alone, ...

**1.5.2 Requisitos de adaptação ao ambiente (Aspectos legais para aderência a legislação, ambiente de operação crítica como indústria, automação, protocolos de comunicação específicos)**

Definem-se aqui possíveis requisitos de adaptação do produto aos ambientes particulares onde ele será implantado. Por exemplo, parâmetros e métodos de configuração requeridos para ambientes específicos devem ser descritos aqui.

| Número de ordem | Requisito | Detalhes |
| --- | --- | --- |
| 1 | Configuração de ticket de venda e da Nota Fiscal  Eletrônica | Configuração dos campos de formulário com interface responsiva. |

***1.6 Funções do produto (Funções básicas, R1.1 ..., R1.2 ...)*** Identificam-se aqui as principais funções que o produto desempenha, descrevendo de forma sintética o objetivo de cada uma.

Requisitos Básicos (O que o sistema deve fazer):

R1 - Controlar a produção

R2 - Atender a demanda de pedidos

R3 - Controlar escalonamento

R4 - Controlar o gerenciamento de estoques

R5 - Controlar a materia prima

R6 - Controlar a entrega de concreta

R7 - Controlar as compra de materia prima

R8 - Controlar o processo de vendas

Estudo de caso baseado no artigo de modelagem de sistemas industriais, Palazzo e Edelweizz (1998), apud Lutherer E. et al., 1994.

O estudo representa uma fábrica de concreto que inclui vendas, escritório e a planta industrial. A planta é composta de vários receptáculos de armazenagem de cimento, areia, cascalho e água; um misturador para preparação do concreto; duas balanças para pesar os componentes; uma válvula de medição de componentes; controle de descarga de água; duas esteiras de transporte para levar os ingredientes sólidos ao misturador.

Os clientes preenchem uma ordem de pedido entregue pelo representante com o tipo e quantidade de concreto para entrega em um ou mais dias pela transportadora da empresa. O departamento de vendas escalona a ordem e eventualmente encaminha a indústria para o encarregado de produção. A indústria providencia a execução do pedido, através da área de produção e de matéria prima. A produção executa a ordem solicitando materiais que são entregues no local pelo encarregado do almoxarifado e estocagem. O operador das máquinas executa os controles de fabricação tendo como auxiliares os operários responsáveis pela mistura dos componentes e pelo funcionário encarregado do controle de fluxo de trabalho. Caso algum produto da matéria prima fique abaixo do nível de segurança, o encarregado da produção emite ordem para o departamento de compras providenciar o pedido da mercadoria que após a entrega é verificado pelo setor de recebimento e distribuição se está entregue conforme foi feito o pedido. Após a fabricação, o produto é entregue na construção e o cliente efetua o pagamento que é recebido pelo departamento financeiro da indústria.

***1.7 Características dos usuários (identificação da formação dos usuários para suas especialidades no uso do sistema,***

***tal como ensino-médio, graduação, especialista, engenheiro, doutor, etc...)***

Descrevem-se aqui as principais características dos grupos de usuários esperados para o produto, tais como cargo ou função, permissão de acesso, frequência de uso, nível de instrução, proficiência no processo de negócio e proficiência em informática.

***1.8 Restrições (Lei Geral de proteção de dados, etc...)*** Descrevem-se aqui aspectos técnicos e gerenciais que possam limitar as opções dos desenvolvedores, tais como restrições legais.

***1.9 Hipóteses de trabalho (Sistema operacional, versão de ferramentas de software, licenças de bibliotecas e de subs sistemas)***

Descrevem-se aqui fatores que não são restrições limitativas do desempenho, como na subseção anterior, mas fatores cuja alteração requer modificações na ER, como, por exemplo, versão a ser utilizada do ambiente operacional ou plataforma de desenvolvimento.

**2 Requisitos específicos**

***2.1 Interfaces externas***

**2.1.1 Visão geral (Tipos de interface, relatórios, gráficos, dashboards)**

Descreve-se aqui, de forma detalhada, todas as entradas e saídas do produto.

**2.1.2 Requisitos para interfaces gráficas de usuário (markups/wireframes)**

Sugere-se, no caso de interfaces gráficas, a inclusão dos seguintes elementos:

● Um esboço do layout gráfico sugerido para a interface;

● Uma descrição dos relacionamentos com outras

interfaces;

● Um diagrama de estados/atividades, caso necessário para

melhor entender-se o comportamento requerido da

interface;

● Uma lista dos campos de dados da interface;

● Uma lista dos comandos da interface;

● BPM;

***2.2 Requisitos funcionais***

**2.2.1 Diagramas de casos de uso (Modelo UML de Casos de Uso)**

Incluir todos os casos de uso que se pretende implementar em uma liberação. Pode-se incluir ainda: um certo caso de uso e seus relacionamentos, todos os casos de uso para um certo ator.

@startuml

left to right direction

skinparam packageStyle rect

actor "operator de valvula de medicao" as Ope\_Va

actor "Operador de mistura" as Ope\_Mi

actor Cliente

actor "Departamento de Vendas" as dv

actor "Encarregado producao" as En\_Pro

actor Produção

actor "Almoxarifado e Estocagem" as ae

actor "Operador de maquinas" as Ope\_Ma

actor "Encarregado de fluxo" as En\_Fl

actor "Setor de Recebimento e Distribuição" as srd

actor Transportadora

actor "Departamento Financeiro" as df

rectangle "Sistema de Produção da Indústria de Concreto" {

usecase "controla decarga de agua" as UC7

usecase "Manipular o misturador" as UC8

usecase "Registrar Pedido" as UC1

usecase "Escalonar Pedido" as UC2

usecase "Controlar Produção" as UC3

usecase "Gerenciar Estoque" as UC4

usecase "Controlar a fabricacão" as UC10

usecase "Controlar o fluxo do trabalho" as UC11

usecase "Emitir ordem para departamento de compras" as UC9

usecase "Receber Pagamento" as UC6

usecase "Gerenciar Transporte e Entrega" as UC5

Cliente -- UC1

UC1 -- dv

dv -- UC2

Ope\_Va -- UC7

Ope\_Mi -- UC8

En\_Pro -- UC9

Ope\_Ma -- UC10

En\_Fl -- UC11

UC3 -- ae

UC4 -- Produção

srd -- UC4

Transportadora -- UC5

UC6 -- df

UC8 .> UC7 : <<incluide>>

UC1 .> UC8 : <<extende>>

UC2 .> UC1 : <<include>>

UC2 <. UC3 : <<extende>>

UC4 .> UC3 : <<include>>

UC10 .> UC4 : <<include>>

UC11 .> UC10 : <<include>>

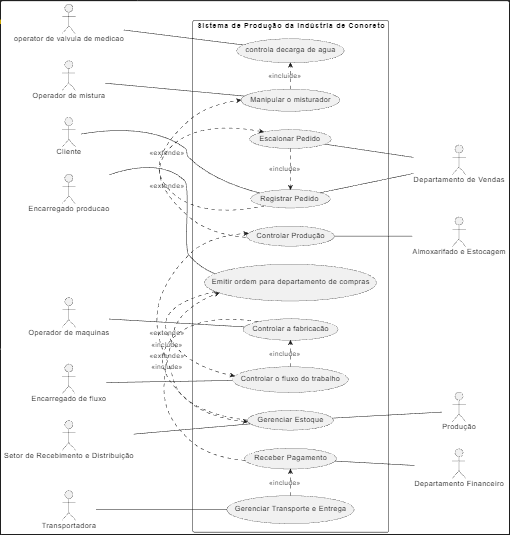
UC9 .> UC11 : <<extende>>

UC6 .> UC9 : <<extende>>

UC5 .> UC6 : <<include>>

}

@enduml



**2.2.2 Fluxos dos casos de uso (Casos de Uso**

**Expandidos e Diagramas de Atividades)**

**Casos de Uso Expandidos**

Caso de uso: Receber Pagamento

Actor primario: Departamento Financiero

Outros atores:Cliente

Descrição: Após o cliente efetuar o pagamento que é recebido pelo departamento financeiro da indústria. O setor de recebimento verifica a entrega e distribuição se está entregue conforme foi feito o pedido

Fluxo Principal:

Ação do Ator Ação do Sistema

1- Departamento Financiero: Processar o Pagamento 2- Notificar o cliente para fazer o pagamento

3- Cliente: Iniciar o pagamento

4- Departamento Financiero: Solicitar a forma de pagamento 5- Mandar as formas de pagamento

6- Cliente: Escolher a forma de pagamento

7- Cliente: Finalizar o pagamento 8- Notificar al departamento

Financiero

9- Departamento Financiero: Receber o Pagamento e Mandar Recibo

10- Departamento Financiero: Verificar a entregar

Fluxo Alternativo:

Acao do Ator Acao do Sistema

1- Departamento Financiero: Verificar se o valor nao foi errado

2- Departamento Financiero: Verificar se o cliente nao pediu ajuda

----------------------------------------------------------------------------------------------

Caso de Uso: Escalonar Pedido

Ator primarío: Departamento de vendas

Descricão: O departamento de vendas escalona a ordem do pedido e eventualmente encaminha a indústria para o encarregado de produção.

Fluxo Principal:

Acao do Ator Acao do Sistema

1- Departamento de vendas: Receber pedido de concreto

2- Departamento de vendas: Escalonar pedido

3- Departamento de vendas: Encaminhar pedido ao Encarregado de Produção 4- Notificar o Encarregado

Fluxo Alternativo:

Acao do Ator Acao do Sistema

1- Controlar o encaminhamento

**Diagramas de Atividades caso de uso receber pagamento**

@startuml

|Departamento Financiro|

start

:Processar o Pagamento;

|Sistema|

:Notificar o cliente para fazer o pagamento;

|Cliente|

:Iniciar o pagamento;

:Solicitar a forma de pagamento;

|Departamento Financiro|

:Mandar as formas de pagamento;

|Cliente|

:Escolher a forma de pagamento;

:Fazer o pagamento;

|Sistema|

:Notifigcar o departamento Financiero;

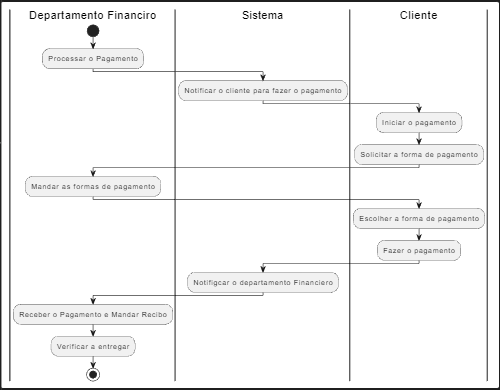
|Departamento Financiro|

:Receber o Pagamento e Mandar Recibo;

:Verificar a entregar;

stop

@enduml



**Diagramas de Atividades do caso de uso escalonar pedido**

@startuml

|Departamento De Venda|

start

:Receber pedido;

:Escalonar pedido;

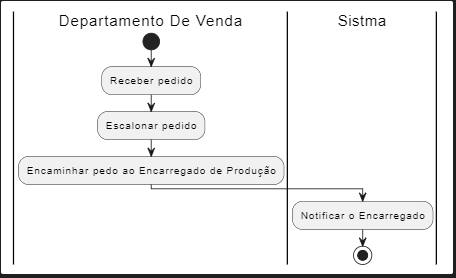
:Encaminhar pedo ao Encarregado de Produção;

|Sistma|

:Notificar o Encarregado;

stop

@enduml



***2.3 Requisitos não-funcionais***

**2.3.1 Requisitos de desempenho (Velocidade de banda, tempo de resposta das interfaces e/ou impressão de relatórios)** Requisitos de desempenho devem ser especificados de forma quantitativa e mensurável.

**2.3.2 Requisitos de dados persistentes (Sistemas de gerenciamento de banco de dados e modelo de**

**persistência)**

Descrevem-se aqui estruturas lógicas de dados persistentes (que mantém seu valor após a execução do programa) que sejam usadas pelo produto. Cada estrutura de dados pode ser, por exemplo, um arquivo convencional ou uma tabela em um banco de dados.

INCLUIR AQUI O MODELO DE BANCO DE DADOS

**2.3.3 Restrições ao desenho**

Restrições de projeto impostas por padrões externos, com influência da legislação, tipo legislação fiscal municipal, estadual e federal.

**2.3.4 Atributos de Qualidade**

Indica os atributos de qualidade, seguindo as características e subcaracterísticas recomendadas pela norma ISO-9126.

***2.4 Objetos/Classes***

**2.4.1 Modelo Conceitual/Classes de Análise/Modelo de Domínio (Classes, Associações, nomes das associações,**

**Multiplicidades e Atributos)**

@startuml

class Fabrica{

- nome: String

- endereco: String

- telefone: String

}

class Funcionario{

- nome: string

- id: string

- tipo: string

}

class Venda{

Departamento: string

idCliente: string

tipoProduto: string

}

class Escritorio{

- nomeSecretatia: string

- nomeSupervisor: string

}

class Planta{

- tipoReceptaculo: string

- tipoMistirador: string

- tipoValvula: string

}

class Cliente{

- id: int

- nome: String

- endereco: String

- telefone: String

}

class Pedido{

- tipoPedido: String

- quantidade: int

- nomeTransportador: string

}

class DepartVenda{

- nomeFuncionario: string

- tipoFuncionario: string

+ escalonarPedido: void

}

class EncarProducao{

- id: int

- nome: string

- endereco: string

}

class Producao{

- tipoProduto: string

- nomeEncarProducao: string

- nomeOperadorMaquina: string

- nomeEncarAlmoxarifado: string

}

class OperMaquina{

- nome: string

- id: int

- tipoMaquina: string

}

class EncarAlmoxarifado{

- nome: string

- id: int

- tipoMaterial: string

}

class DepartCompra{

- nomeFuncionario: string

- tipoFuncionario: string

}

class SetorRecebimento{

- nomeFuncionario: string

- tipoFuncionario: string

}

class DepartFinanciera{

- nomeFuncionario: string

- tipoFuncionario: strings

}

Venda "1"--\* Fabrica

Planta "1"--"1" Fabrica

Escritorio "1"--"1" Fabrica

Escritorio "1"--"1" Funcionario

Planta "1"--"1" Producao

Venda "1"--"1" Cliente

Venda "1"--"1" DepartVenda

Venda "1"--"1" DepartCompra

Venda "1"--"1" DepartFinanciera

Venda "1"--"1" SetorRecebimento

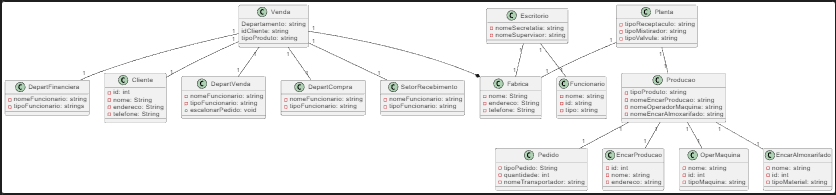
Producao "1"--"1" OperMaquina

Producao "1"--"1" EncarAlmoxarifado

Producao "1"--"1" EncarProducao

Producao "1"--"1" Pedido

@enduml



**2.4.2 DSS – Diagramas de Sequência do Sistema (Eventos e Operações) de Casos de Uso (denotando as mensagens entre os objetos do domínio para atender ao Caso de Uso).**

@startuml

"Departamento Financiero" --> Cliente: processarPagamento(idCliente, nomeCliente, valor)

**Diagramas de Sequência do Sistema caso de uso receber pagamento**

Cliente <-- Sistema: notificarClente()

Cliente --> Sistema:iniciarPagamento(id, senha)

"Departamento Financiero" --> Cliente: solicitarFormaPagamento()

Cliente <-- Sistema: mandararFormaPagamento(forma)

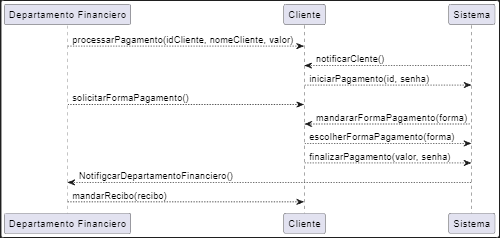
Cliente --> Sistema: escolherFormaPagamento(forma)

Cliente --> Sistema: finalizarPagamento(valor, senha)

"Departamento Financiero" <-- Sistema: NotifigcarDepartamentoFinanciero()

"Departamento Financiero" --> Cliente: mandarRecibo(recibo)

@enduml



**Diagramas de Sequência do Sistema do caso de uso escalonar pedido**

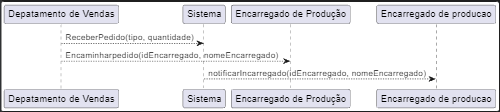
@startuml

"Depatamento de Vendas" --> Sistema: ReceberPedido(tipo, quantidade)

"Depatamento de Vendas" --> "Encarregado de Produção": Encaminharpedido(idEncarregado, nomeEncarregado)

Sistema --> "Encarregado de producao": notificarIncarregado(idEncarregado, nomeEncarregado)

@enduml



**2.4.3 Contratos (das Operações do DSS)**

Nome: processarPagamento(idCliente, nomeCliente, valor)

**Contrato do diagramas de Sequência do Sistema caso de uso receber pagamento**

caso de uso: Receber pagamento

Descrição: Após a fabricação, o produto é entregue

na construção e o departamento financeiro da indústria recebe

o pagamento que é efetuado pelo cliente.

Referencias: R8

Pré-condições: O cliente fazer o pedido escalonado pelo

departamento de vendas.

Post-condições: O cliente inicia o pagamento, colocando seu id

e sua senha.

Nome: iniciarPagamento(id, senha)

caso de uso: Receber pagamento

Descrição: O cliende iniciar o pagamento que for processado

pelo departamento fianciero em colocando seu id e sua senha.

Referencias: R8

Pré-condições: O departamento Financiero rocessar o pegamendo

do pedido comprado.

Post-condições: Solicitar a forma de partamento que tem

no sistema.

Nome: solicitarFormaPagamento()

caso de uso: Receber pagamento

Descrição: O cliende solicita as formas de pagamento

disponibilizado para fazer o pagamento.

Referencias: R8

Pré-condições: O cliende iniciar o pagamento que for processado

pelo departamento fianciero em colocando seu id e sua senha.

Post-condições: O sistema mandar as formas de pagamento

diponibilizado.

Nome: mandararFormaPagamento(forma)

caso de uso: Receber pagamento

Descrição: O sistema manda as formas de pagamento que tem para

o cliente.

Referencias: R8

Pré-condições: O cliende solicita as formas de pagamento

disponibilizado para fazer o pagamento.

Post-condições: O cliente escolhe uma forma de pagamento.

Nome: escolherFormaPagamento(forma)

caso de uso: Receber pagamento

Descrição: O cliente escolhe uma forma de pagamento depois sabe

as formas disponivel.

Referencias: R8

Pré-condições: sistema manda as formas de pagamento que tem para

o cliente.

Post-condições: O cliente finilizar o pagamento em colocando

**Contrato do diagramas de Sequência do Sistema do caso de uso escalonar pedido**

@startuml

Nome: ReceberPedido(tipo, quantidade)

caso de uso: Escalonar pedido

Descrição: O departamento de vendas recebe o pedido fez pelo

cliente.

Referencias: R3

Pré-condições: O cliente fazer o pedido.

Post-condições: O departamento de venda escalonar o pedido.

Nome: EscalonarPedido(ordem)

caso de uso: Escalonar Pedido

Descrição: O departamento escalona a ordem do pedido.

Referencias: R3

Pré-condições: O departamento de vendas recebe o pedido fez pelo

cliente.

Post-condições: Eventualmente encaminha a industria para O

encaregado de produção

Nome: Encaminharpedido(idEncarregado, nomeEncarregado)

caso de uso: Escalonar Pedido

Descrição: O departamento de venda encaminha a industria para

o encarregado de produção se for necessario.

Referencias: R3

Pré-condições: O departamento escalona a ordem do pedido.

Post-condições: O sistema notificar o encarregado de produção

sobre o encaminhamento.

Nome: notificarIncarregado(idEncarregado, nomeEncarregado)

caso de uso: Escalonar Pedido

Descrição: O sistema notificar o encarregado de produção

sobre o encaminhamento.

Referencias: R3

Pré-condições: O departamento de venda encaminha a industria para

o encarregado de produção se for necessario.

Post-condições: A indústria providência a execução do pedido,

através da área de produção e de matéria prima.

**2.4.4 Classes de Implementação - Diagrama de Classes (Classes, Associações, nomes das associações, Multiplicidades, Atributos e Métodos). Atribuição de responsabilidades com GRASP (General Responsibility Assignment Software Patterns) que são um conjunto de princípios e diretrizes para atribuição de**

**responsabilidades em projetos de software orientados a objetos.**

**Classes de Implementação - Diagrama de Classes caso de uso receber pagamento**

@startuml

class Cliente{

- nome: string

- id: string

- endereco: string

- contato: string

+ iniciarPagamento(): void

+ solicitarFormaPagamento(): void

+ escolherFormaPagamento(): void

+ fazerPagamento()

}

class Pagamento{

- formaPagamento: string

- dataPagamento: string

}

class DepartamentoFinanciero{

- idRespnsavel: string

- nomeResponsavel: string

+ processarPagamento(): void

+ mandarFormaPagamento(): string

+ receberPagamento(): void

+ verificarEntrega(): void

}

class Entrega{

- dataDaEntrega: string

- enderecoParaEntrega: string

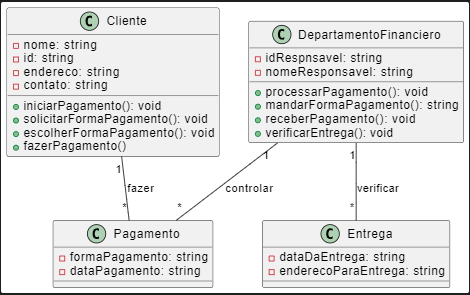
}

Cliente "1" -- "\*"Pagamento: fazer

DepartamentoFinanciero "1" -- "\*"Pagamento: controlar

DepartamentoFinanciero "1" -- "\*"Entrega: verificar

@enduml



**Classes de Implementação - Diagrama de Classes do caso de uso Escalonar Pedido**

@startuml

class DepartamentoVenda{

- idRespnsavel: string

- nomeResponsavel: string

+ receberPedido(): void

+ escalonarPedido(): string

+ Encaminhar(): void

}

class Pedido{

- tipoPedido: String

- quantidade: int

- nomeTransportador: string

+ ordemPedido(): void

}

class EncarProducao{

- id: int

- nome: string

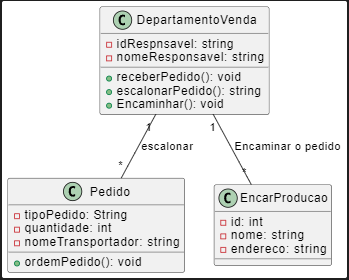
- endereco: string

}

DepartamentoVenda "1" -- "\*" Pedido: escalonar

DepartamentoVenda "1" -- "\*" EncarProducao: Encaminar o pedido

@enduml



**3 Análise de UCP**

As tabelas de escopo de valor do produto e tempo de desenvolvimento com Use Case Points - UCP.

Referências:

*IEEE Std. 830 – 1993. IEEE Recommended Practice for Software Requirements Specifications.*

*IEEE ISO/IEC/IEEE 29148 – 2011. IEEE Systems and software engineering — Life cycle processes — Requirements engineering*

**OBSERVAÇÃO: Os itens deste modelo de especificação, recomendado pela IEEE, poderão ser complementados com novos itens caso sejam justificáveis.**